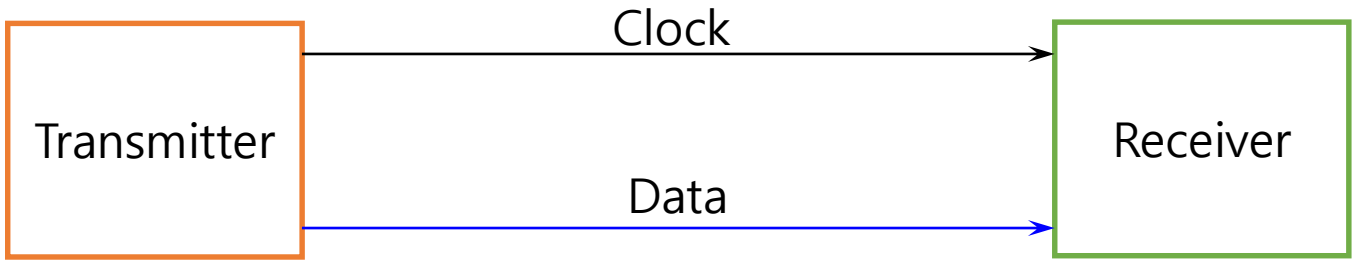
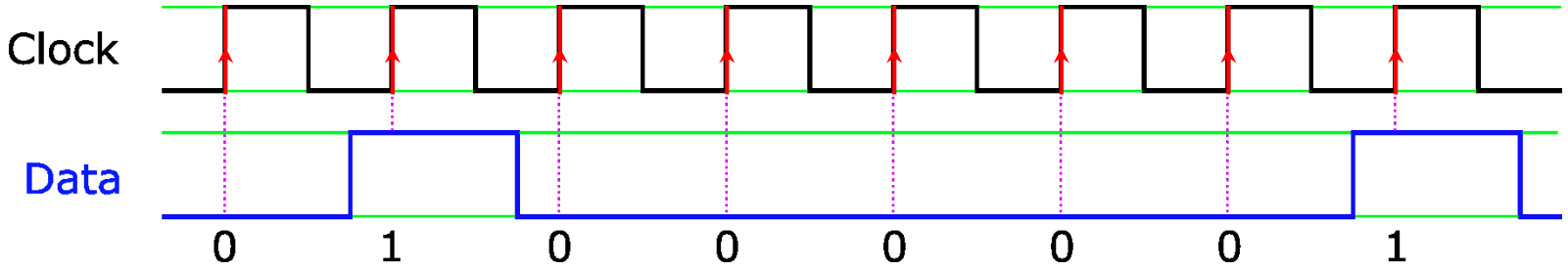
USART

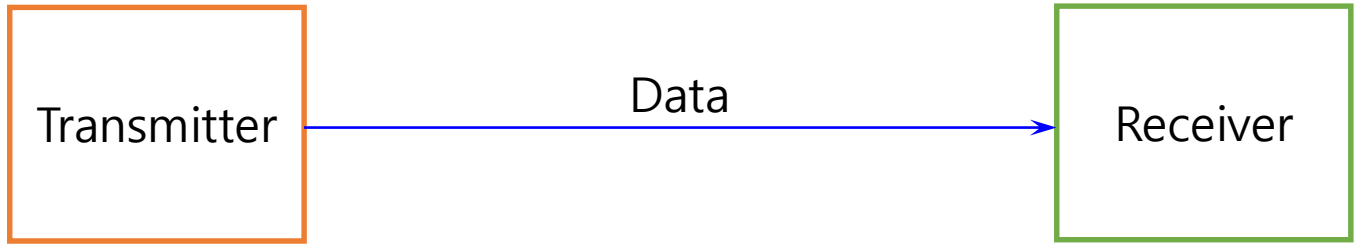
Universal Synchronous/Asynchronous Receiver and Transmitter

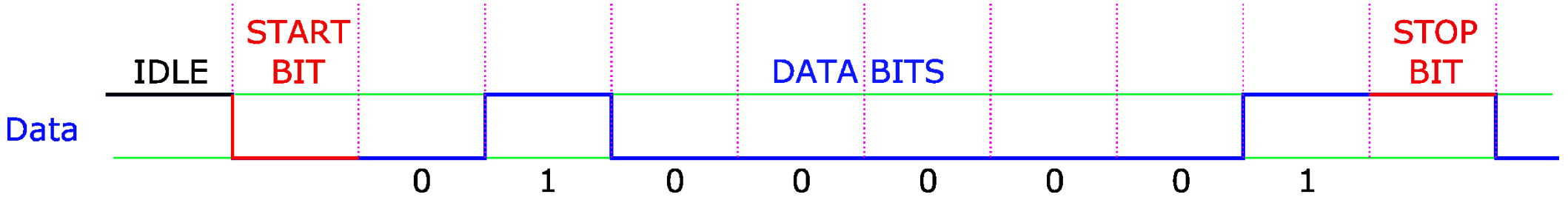
Synchronous Serial Communication



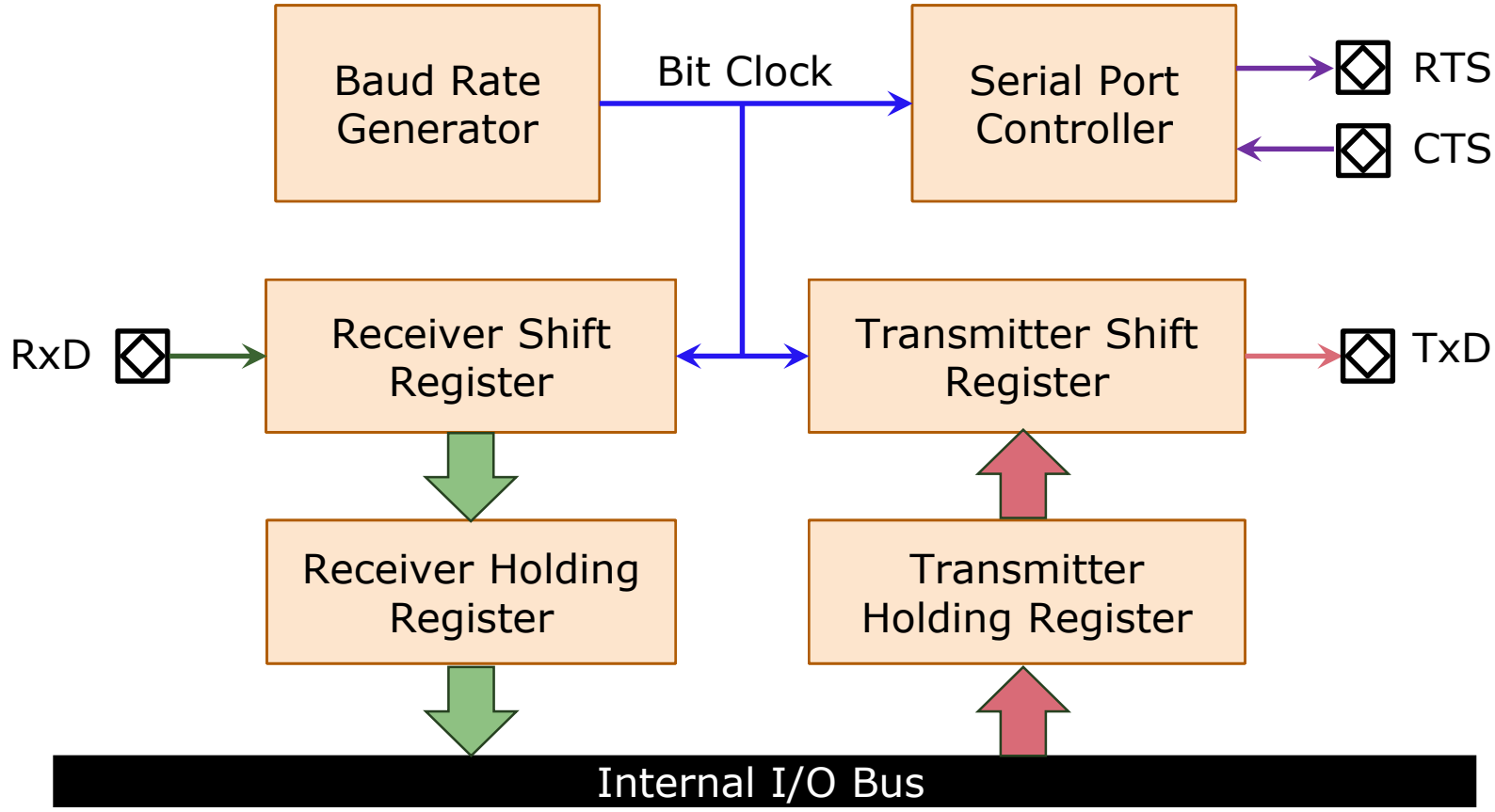


Asynchronous Serial Communication

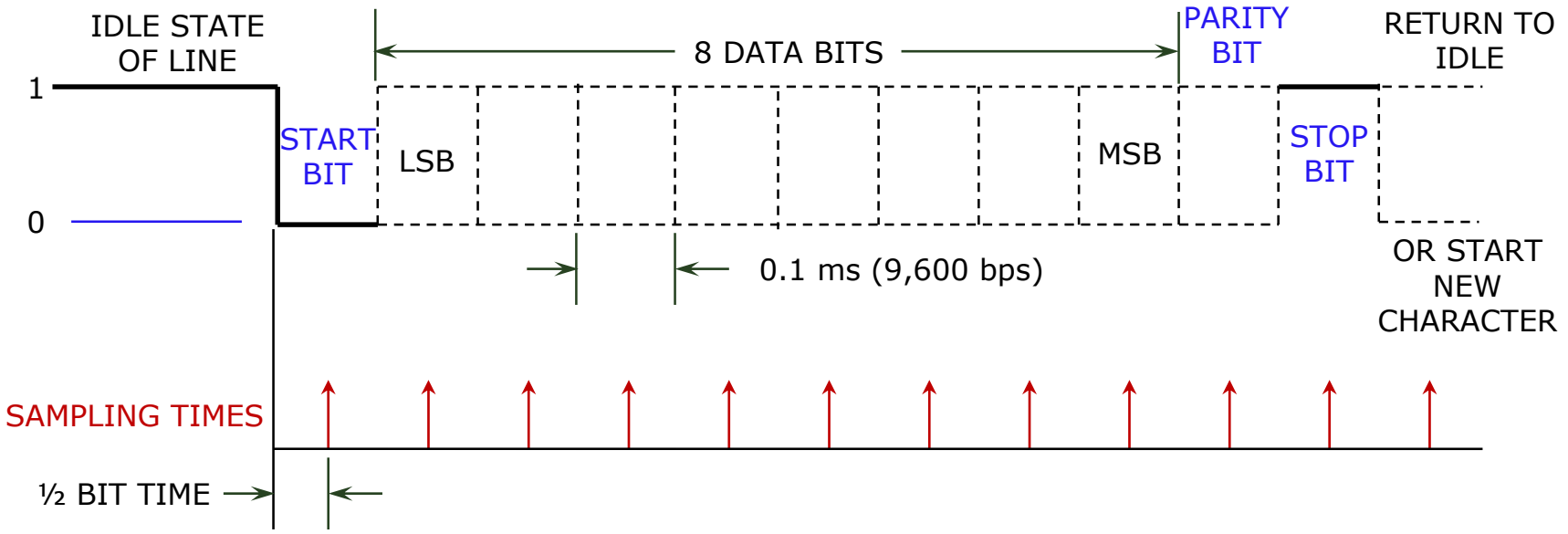


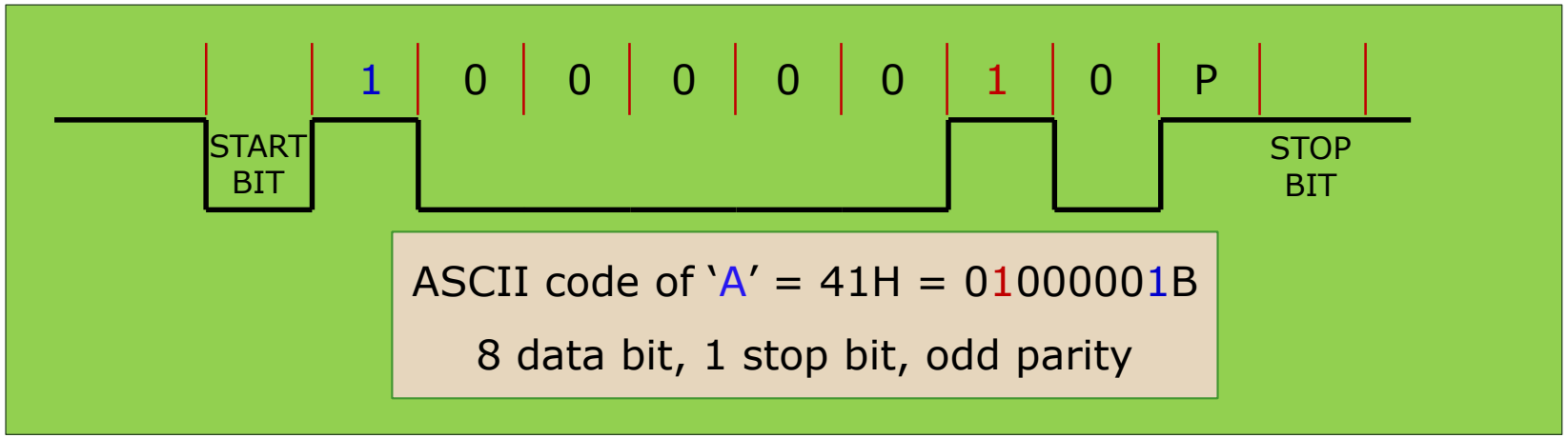


UART Functional Block Diagram



UART Serial Data Transmission





Baud Rate and Bit Rate

- Bit Rate
  - The number of bits conveyed or processed per unit of time.
  - Unit: bits per second (bit/s or bps).
  - can be used interchangeably with "baud" *only* when there are two levels or symbols, representing 0 and 1 respectively.
- Baud Rate
  - The number of distinct symbol changes (signaling events) made to the transmission medium per second.
  - The term baud rate is the same as bit rate when only one bit per symbol is used. (Binary "0" is represented by one symbol, and binary "1" by another symbol)
  - Unit: Bd (/ˈbɔːd/)

UART Special Receiver Conditions (1)

- Overrun error
  - An "overrun error" occurs when the receiver cannot process the character that just came in before the next one arrives.
  - Various devices have different amounts of buffer space to hold received characters.
  - The CPU must service the UART in order to remove characters from the input buffer.
  - If the CPU does not service the UART quickly enough and the buffer becomes full, an Overrun Error will occur, and incoming characters will be lost.

UART Special Receiver Conditions (2)

- Underrun error
  - An "underrun error" occurs when the UART transmitter has completed sending a character and the transmit buffer is empty.
  - In asynchronous modes this is treated as an indication that no data remains to be transmitted, rather than an error, since additional stop bits can be appended.
  - This error indication is commonly found in USARTs, since an underrun is more serious in synchronous systems.

UART Special Receiver Conditions (3)

- Framing error
  - A "framing error" occurs when the designated "start" and "stop" bits are not valid.
  - As the "start" bit is used to identify the beginning of an incoming character, it acts as a reference for the remaining bits.
  - If the data line is not in the expected idle state when the "stop" bit is expected, a Framing Error will occur.

UART Special Receiver Conditions (4)

- Parity error
  - A "parity error" occurs when the number of "active" bits does not agree with the specified parity configuration of the USART, producing a Parity Error.
  - Because the "parity" bit is optional, this error will not occur if parity has been disabled.
  - Parity error is set when the parity of an incoming data character does not match the expected value

RS-232

- RS-232 (Recommended Standard 232)
  - Traditional name for a series of standards for serial binary single-ended data and control signals connecting between a DTE (Data Terminal Equipment) and a DCE (Data Circuit-terminating Equipment).
  - The current version of the standard is *TIA-232-F Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, issued in 1997.

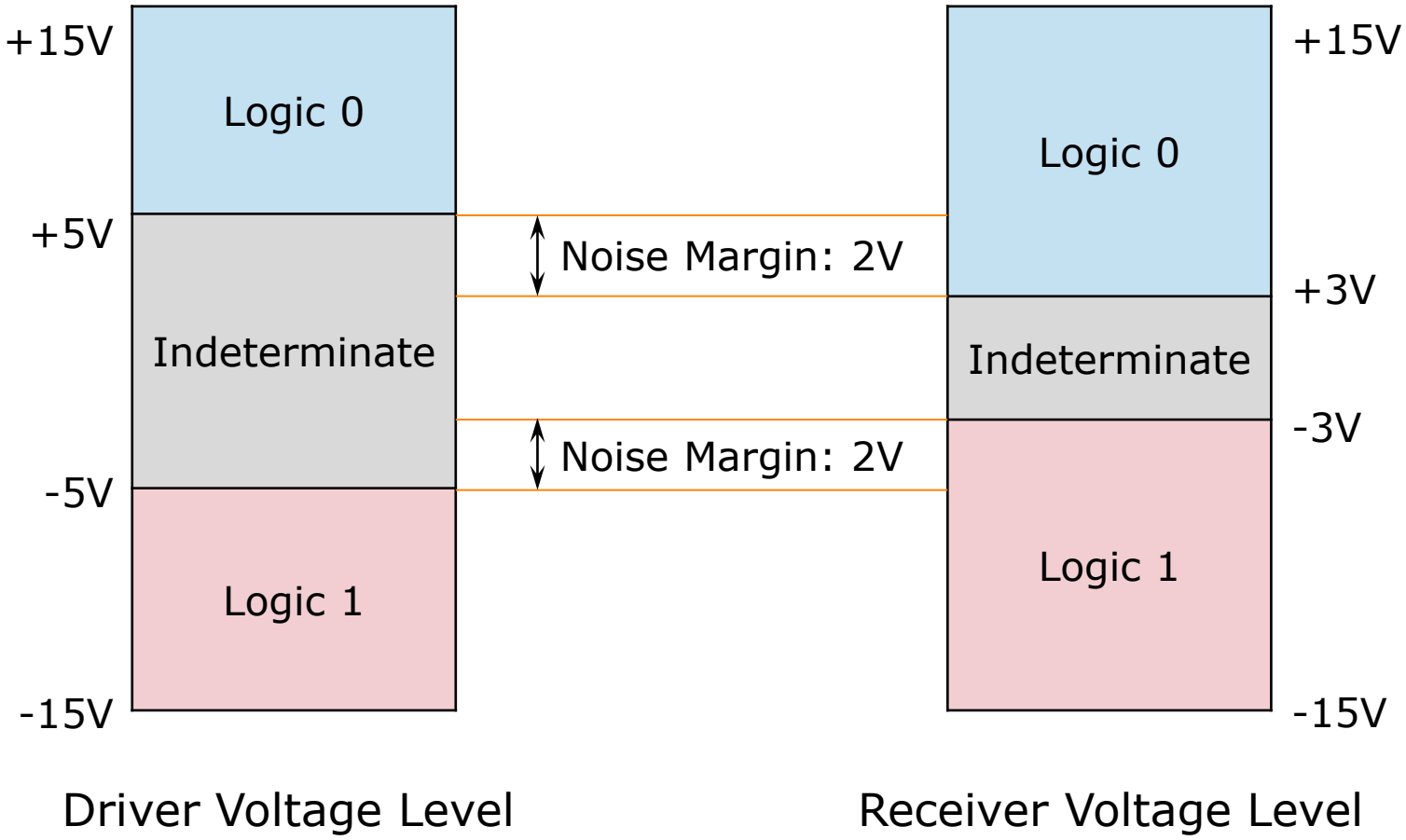
EIA RS-232

- The Electronic Industries Alliance (EIA) standard RS-232-C as of 1969 defines:
  - Electrical signal characteristics such as voltage levels, signaling rate, timing and slew-rate of signals, voltage withstand level, short-circuit behavior, and maximum load capacitance.
  - Interface mechanical characteristics, pluggable connectors and pin identification.
  - Functions of each circuit in the interface connector.
  - Standard subsets of interface circuits for selected telecom applications.

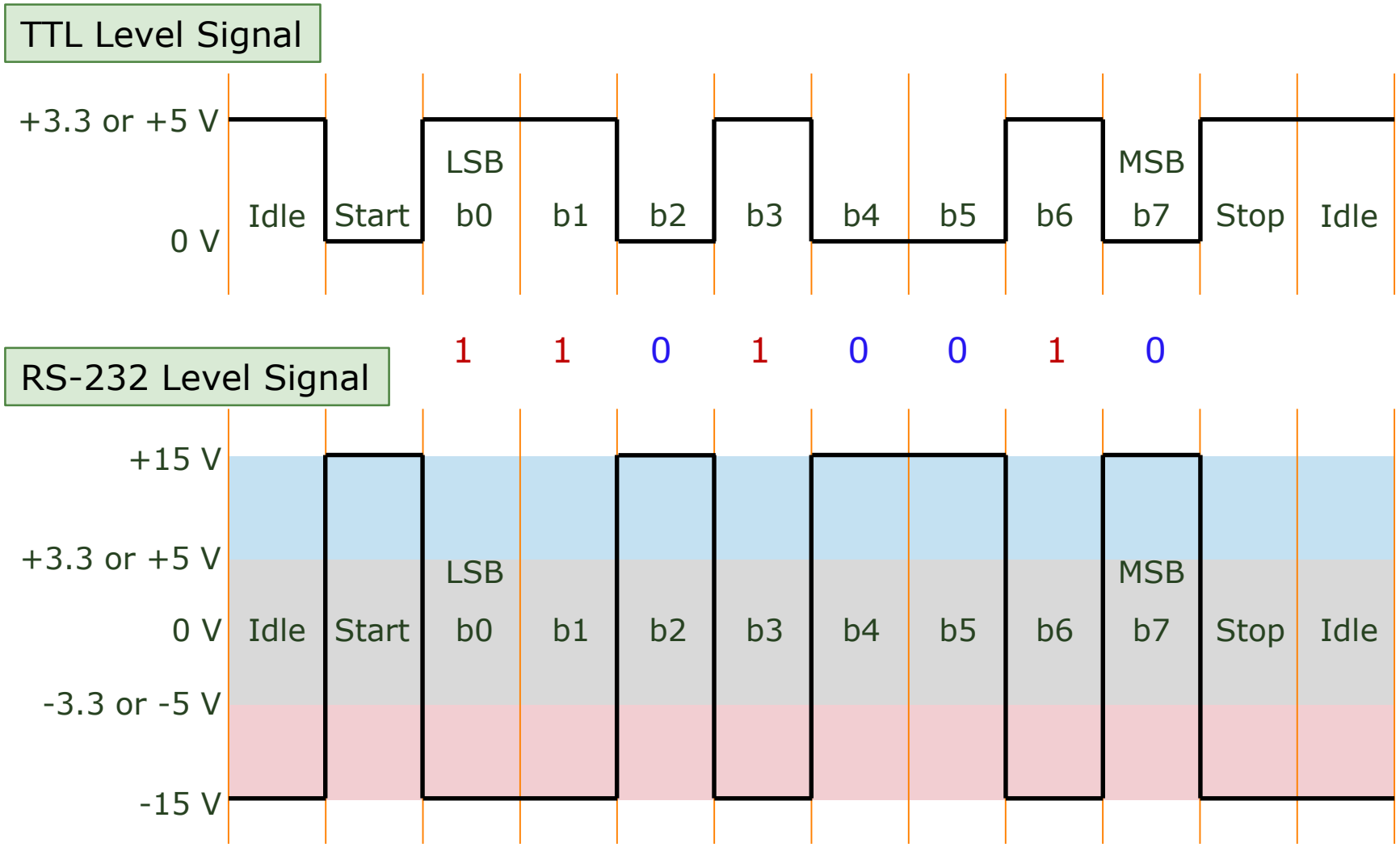
EIA RS-232 Electrical Specifications

SPECIFICATIONS		RS232	RS423
Mode of Operation		Single-Ended	Single-Ended
Total Number of Drivers and Receivers on One Line		1 DRIVER 1 RECVR	1 DRIVER 10 RECVR
Maximum Cable Length		50 FT.	4000 FT.
Maximum Data Rate		20kb/s	100kb/s
Maximum Driver Output Voltage		+/-25V	±6V
Driver Output Signal Level (Loaded Min.)	Loaded	±5V to ±15V	±3.6V
Driver Output Signal Level (Unloaded Max)	Unloaded	±25V	±6V
Driver Load Impedance (Ohms)		3k to 7k	>=450
Max. Driver Current in High Z State	Power On	N/A	N/A
Max. Driver Current in High Z State	Power Off	±6mA @±2v	±100uA
Slew Rate (Max.)		30V/uS	Adjustable
Receiver Input Voltage Range		±15V	±12V
Receiver Input Sensitivity		±3V	±200mV
Receiver Input Resistance (Ohms)		3k to 7k	4k min.

EIA RS-232 Voltage Levels



EIA RS-232 Voltage Levels



EIA RS-232 Wiring

RTS/CTS Handshaking (5-wire RS-232)

Signal	DB-25	DB-9		DB-9	DB-25	Signal
RTS	4	7		8	5	CTS
CTS	5	8		7	4	RTS
TxD	2	3		2	3	RxD
RxD	3	2		3	2	TxD
GND	7	5		5	7	GND

3-wire RS-232

Signal	DB-25	DB-9		DB-9	DB-25	Signal
TxD	2	3		2	3	RxD
RxD	3	2		3	2	TxD
GND	7	5		5	7	GND

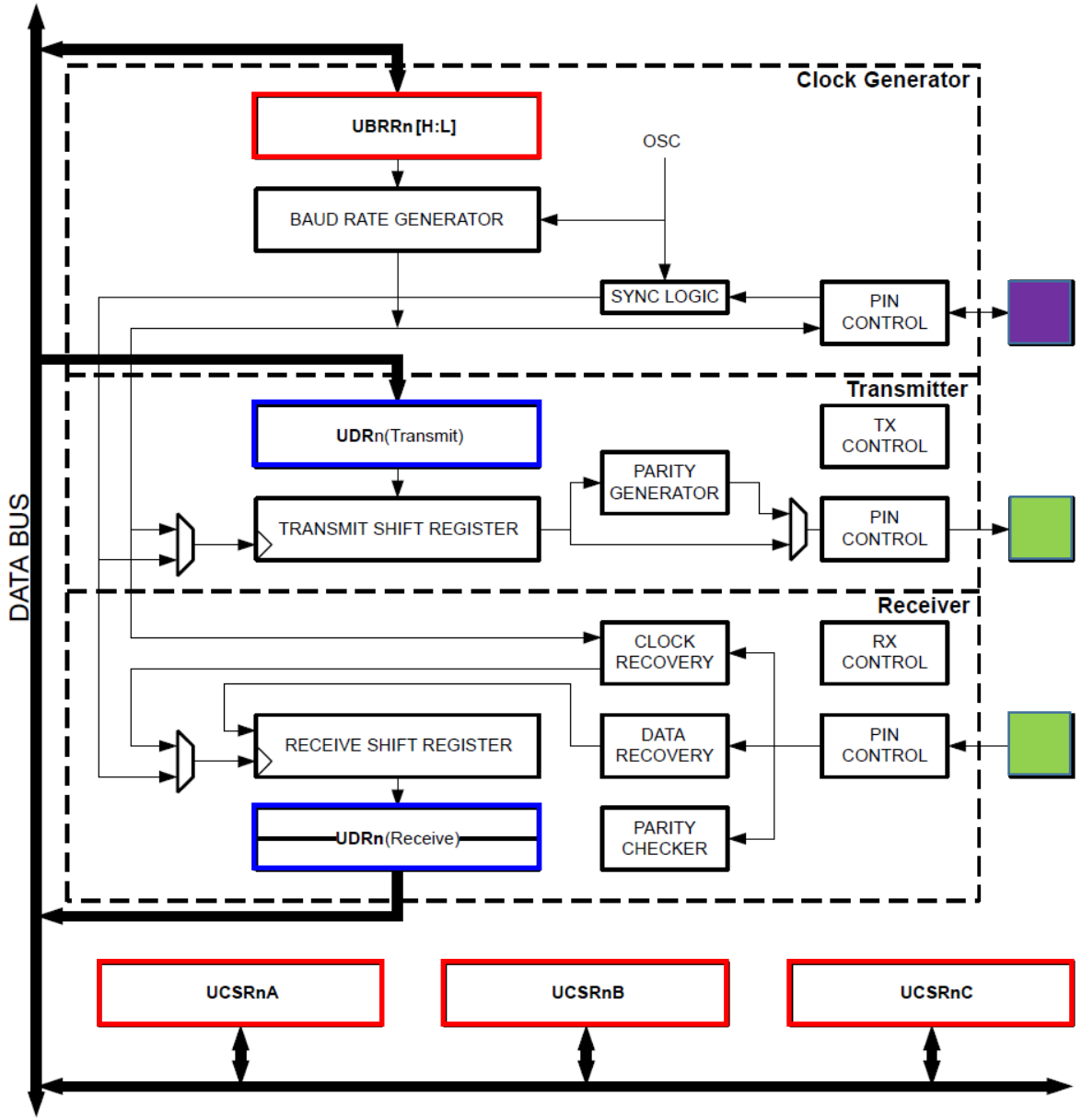
ATmega328PB USART Features (1)

- Two USART instances USART0, USART1
- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data Overrun Detection
- Framing Error Detection

ATmega328PB USART Features (2)

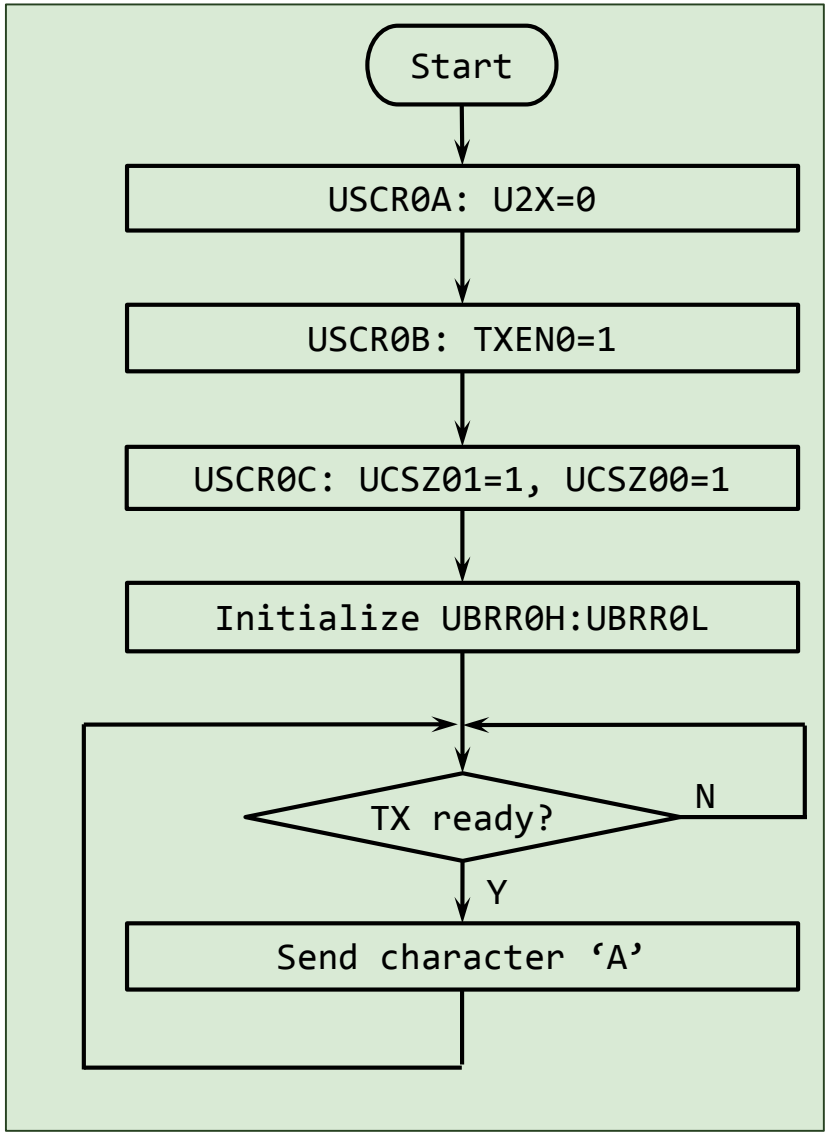
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode
- Start Frame Detection

ATmega328PB USART Block Diagram



ATmega328PB USART0 Example 1 (Polling) (1)

- Specifications:
  - 9,600 baud rate,
  - 8 data bits,
  - no parity,
  - 1 stop bit
- Transmits character ‘A’ via USART0 continuously.
- Use Polling method



ATmega328PB USART0 Example 1 (Polling) (2)

Asynchronous, no parity, 1 stop bit, 8 data bits

UCSR0C

UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0
0	0	0	0	0	1	1	0

Mode of operation
UMSEL01:UMSEL00=00 (Async)

Parity mode
UPM01:UPM00=00 (Disabled)

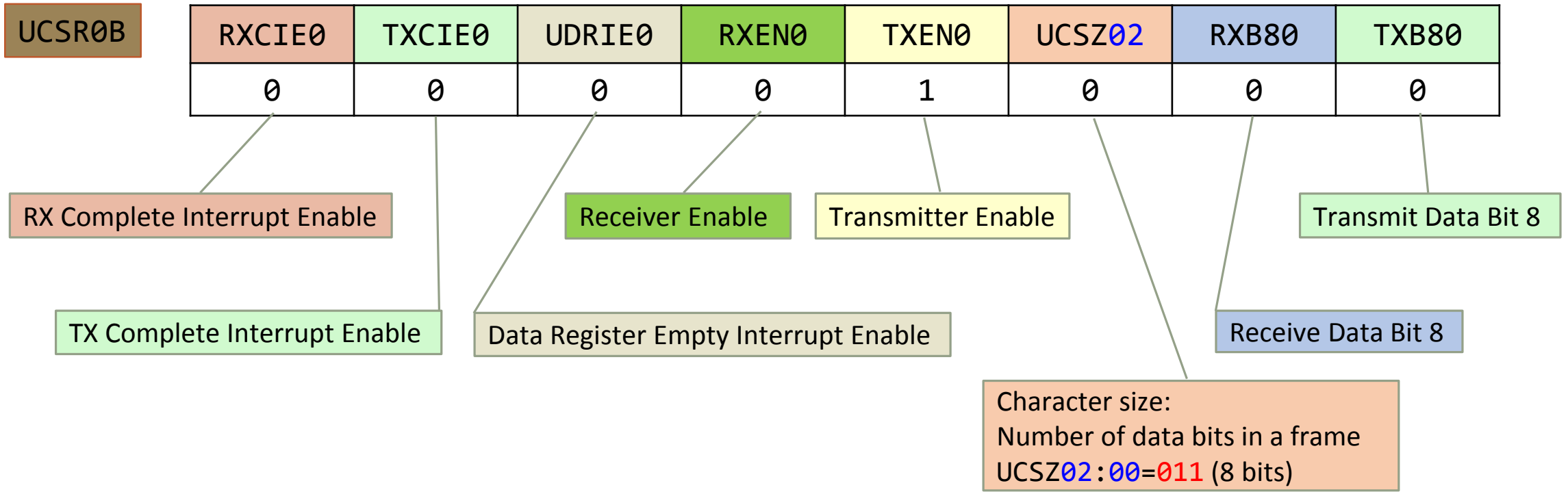
Number of stop bits
USBS0=0 (1 bit)

Character size:
Number of data bits in a frame
UCSZ02:00=011 (8 bits)

Clock Polarity
Synchronous only

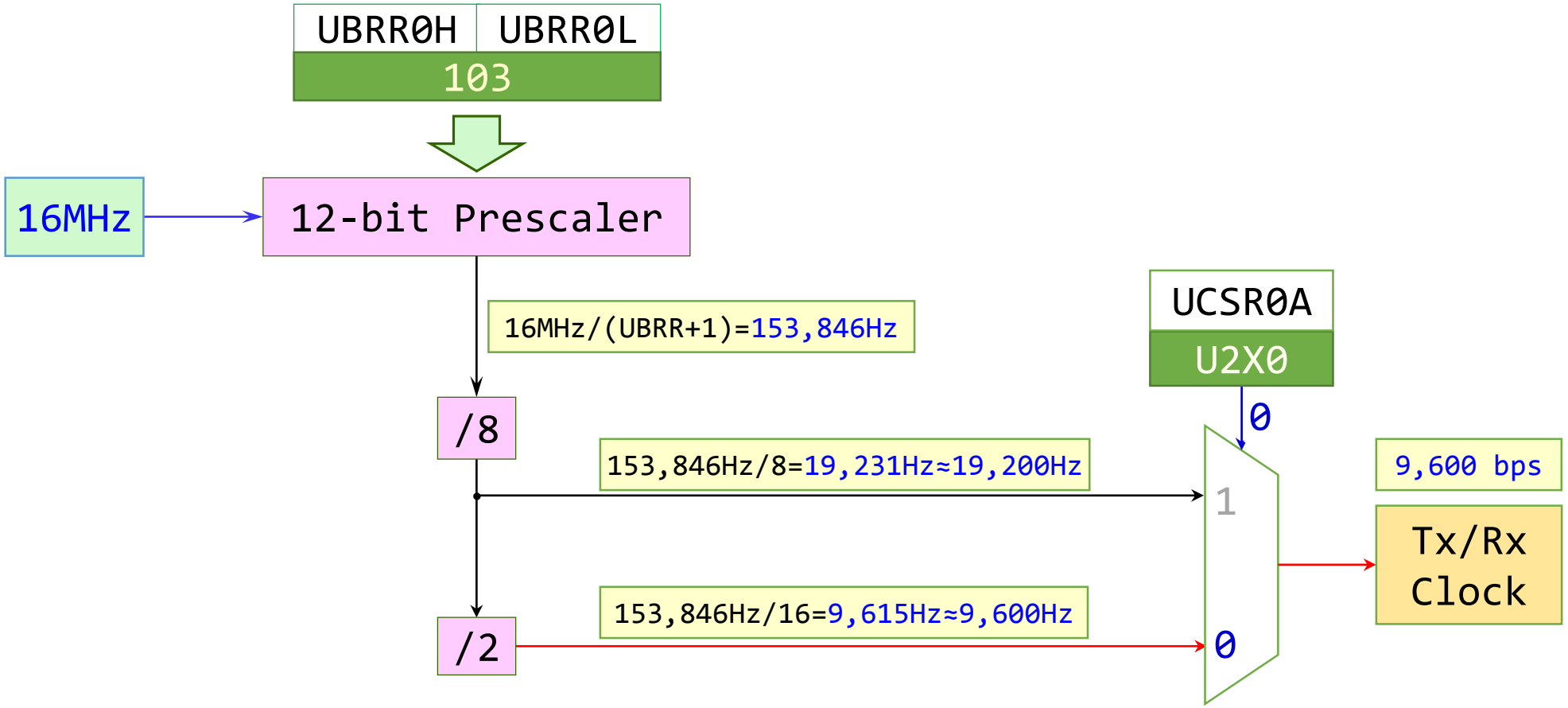
ATmega328PB USART0 Example 1 (Polling) (3)

Polling (non-interrupt), enable TX0



ATmega328PB USART0 Tx/Rx Clock Generation

```
SYSCLK = 16 MHz
Baud Rate = 16,000,000/(103+1)/8/2=9,615Hz≈9,600Hz
```



ATmega328PB USART0 Example 1 (Polling) (4)

```
Baud Rate = 9,600 bps
```

UCSR0A

RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
0	0	0	0	0	0	0	0

- RXC0: RX Complete
- TXC0: TX Complete
- UDRE0: Data Register Empty
- FE0: Frame Error
- DOR0: Data Overrun
- UPE0: Parity Error
- U2X0: Double TX Speed
- MPCM0: Multi-processor Communication Mode

UBRR0H	UBRR0L
0	103

$$UBRR0 = \frac{f_{osc}}{16 \times BAUD} - 1 \qquad \text{(when U2X0=0)}$$

ATmega328PB USART0 Example 1 (Polling) (5)

- Specifications:
  - 9,600 baud rate,
  - 8 data bits,
  - no parity,
  - 1 stop bit
- Transmits character ‘A’ via USART0 continuously.
- Use Polling method

```
#include <avr/io.h>

int main(void)
{
    UCSR0A = 0b00000000;  // U2X0=0: No double speed
    UCSR0B = 0b00001000;  // Enable Tx, 8 Data bits
    UCSR0C = 0b00000110;  // Async mode, No Parity,
                          // 1 Stop bit, 8 Data bits
    UBRR0  = 103;         // Baud Rate=16MHz/(9600*16) - 1

    while (1)
    {
        // Wait until Tx Data Register Empty
        while ((UCSR0A & 0b00100000) == 0);

        UDR0 = 'A';      // Transmit character ‘A’
    }
}
```

ATmega328PB USART0 Example 1 (Polling) (6)

- Specifications:
  - 9,600 baud rate,
  - 8 data bits,
  - no parity,
  - 1 stop bit
- Transmits character ‘A’ via USART0 continuously.
- Use Polling method

Same program with different format

```
#define F_CPU  16000000UL
#define UART_BAUD_RATE  9600UL
#define DIVISOR(((F_CPU / (UART_BAUD_RATE * 16UL))) - 1)

#include <avr/io.h>

int main(void)
{
    UCSR0A &= ~(1 << U2X0);          // U2X0=0: No double speed
    UCSR0B = 1 << TXEN0;             // Enable Tx, 8 Data bits
    UCSR0C = (0b00 << UMSEL00)       // Async mode
           | (0b00 << UPM00)         // No Parity
           | (0 << USBS0)            // 1 Stop bit
           | (0b11 << UCSZ00);       // 8 Data bits
    UBRR0  = DIVISOR;                // Baud Rate

    while (1)
    {
        while ((UCSR0A & (1 << UDRE0)) == 0);  // Wait until Tx Data Register Empty

        UDR0 = 'A';  // Transmit character ‘A’
    }
}
```

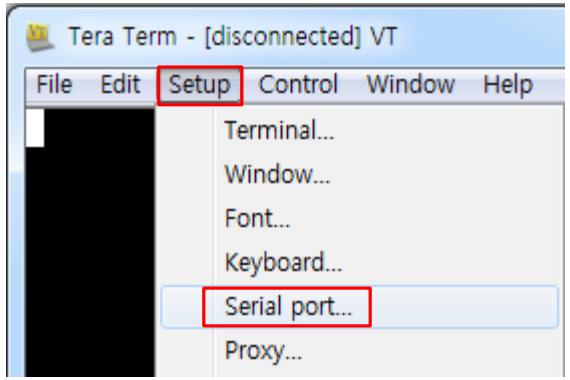
ATmega328PB USART0 Example 1 (Polling) (7)

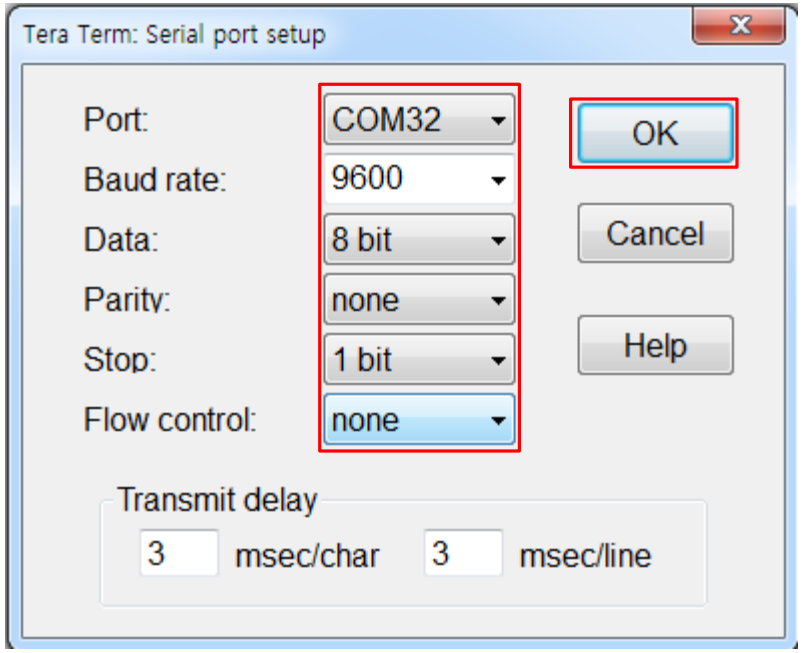
Same program with functions

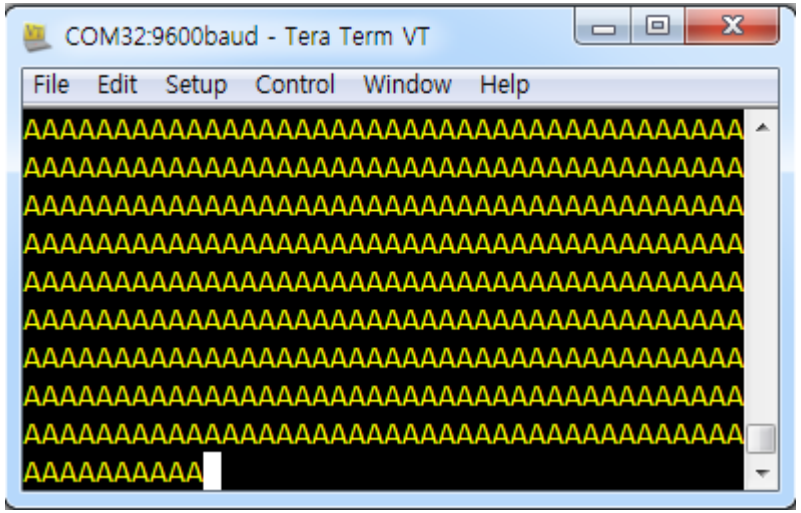
```
#define F_CPU  16000000UL
#define UART_BAUD_RATE  9600UL
#define DIVISOR(((F_CPU/(UART_BAUD_RATE*16UL)))-1)

#include <avr/io.h>

void uart0_init(void);
void uart0_putchar(char ch);

int main(void)
{
    uart0_init();

    while (1)
    {
        uart0_putchar('A');
    }
}
```

```
void uart0_init(void)
{
    UCSR0A &= ~(1 << U2X0); // U2X0=0: No double speed
    UCSR0B = 1 << TXEN0;    // Enable Tx, 8 Data bits
    UCSR0C = (0b00 << UMSEL00)  // Async mode
           | (0b00 << UPM00)      // No Parity
           | (0 << USBS0)         // 1 Stop bit
           | (0b11 << UCSZ00);    // 8 Data bits
    UBRR0  = DIVISOR;             // Baud Rate
}

void uart0_putchar(char ch)
{
    // Wait until Tx Data Register Empty
    while ((UCSR0A & (1 << UDRE0)) == 0);

    UDR0 = ch;
}
```

ATmega328PB USART0 Example 1 (Polling) (8) (Tera Term)







ATmega328PB USART0 Example 2 (Polling) (1)

- Specifications:
  - baud rate: 1 Mbps, 8 data bits, no parity, 1 stop bit
- Print out ‘Switch pressed.’ message when the SWITCH at PB7 is pressed and ‘Switch released.’ message when the SWITCH 7 is released .
- Use Polling method

ATmega328PB USART0 Example 2 (Polling) (2)

- The following statements must be included in the source to use `printf()` function:

```
FILE uart_dev1 = FDEV_SETUP_STREAM(uart0_putchar, NULL, _FDEV_SETUP_WRITE);
stdout = &uart_dev1;
```

- The following statements must be included in the source to use `scanf()` function:

```
FILE uart_dev2 = FDEV_SETUP_STREAM(NULL, uart0_getchar, _FDEV_SETUP_READ);
stdin = &uart_dev2;
```

ATmega328PB USART0 Example 2 (Polling) (3)

```c
#include <avr/io.h>
#include <stdio.h>

FILE uart_dev1 = FDEV_SETUP_STREAM(uart0_putchar, NULL, _FDEV_SETUP_RW);

void init_board(void)
{
	// 8 Data, 1 Stop, No Parity, Baud Rate: 1 Mbps
	UCSR0A=0x00;
	UCSR0B=0x18;
	UCSR0C=0x06;
	UBRR0H=0x00;
	UBRR0L=0;   // 1 Mbps
}

void uart0_putchar(char ch)
{
	while (!(UCSR0A & (1 << UDRE0))); // Wait for empty transmit buffer
	UDR0 = ch;  // Put data into buffer, sends the data
}
```

```c
int main(void)
{
	unsigned int count = 0;

	init_board();

	stdout = &uart_dev1;
	printf("Hello, I'm an ATmega328P X PLAINED MINI.\n");


	while (1)
	{
		if ((PINB & 0x80) == 0)// SWITCH pressed
		{
			printf("Switch pressed (%d).\n", count);
			count++;
			while ((PINB & 0x80) == 0);
		}
		else
		{
			printf("Switch released.\n");
			while ((PINB & 0x80) != 0);
		}
	}
}
```

USART0

END